Scheduling Continuous Operators for IoT Edge Analytics

Patient Ntumba, Nikolaos Georgantas, Vassilis Christophides

▶ **To cite this version:**

Patient Ntumba, Nikolaos Georgantas, Vassilis Christophides. Scheduling Continuous Operators for IoT Edge Analytics. EdgeSys '21 - 4th International Workshop on Edge Systems, Analytics and Networking colocated with EuroSys'21, Apr 2021, Online United Kingdom, United Kingdom. pp.55-60, 10.1145/3434770.3459738 . hal-03208518v2

HAL Id: hal-03208518
https://hal.inria.fr/hal-03208518v2

Submitted on 7 Jun 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Scheduling Continuous Operators for IoT Edge Analytics

Patient Ntumba
patient.ntumba@inria.fr
Inria Paris, France

Nikolaos Georgantas
nikolaos.georgantas@inria.fr
Inria Paris, France

Vassilis Christophides
Vassilis.Christophides@ensea.fr
ENSEA, ETIS, France

ABSTRACT

In this paper we are interested in exploring the Edge-Fog-Cloud architecture as an alternative approach to the Cloud-based IoT data analytics. Given the limitations of Fog in terms of limited computational resources that can also be shared among multiple analytics with continuous operators over data streams, we introduce a holistic cost model that accounts both the network and computational resources available in the Edge-Fog-Cloud architecture. Then, we propose scheduling algorithms RCS and SOO-CPLEX for placing continuous operators for data stream analytics at the network edge. The former dynamically places continuous operators between the Cloud and the Fog according to the evolution of data streams rates and uses as less as possible Fog computational resources to satisfy the constraints regarding the usage of both computational and network resources. The latter statically places continuous operators between the Cloud and the Fog to minimize the overall computational and network resource usage cost. Based on thorough experiments, we evaluate the effectiveness of SOO-CPLEX and RCS using simulation.

CCS CONCEPTS

• **Information systems** → **Stream management**; • **Networks** → *Network management*; *Cloud computing*.

ACM Reference Format:
Patient Ntumba, Nikolaos Georgantas, and Vassilis Christophides. 2021. Scheduling Continuous Operators for IoT Edge Analytics. In *4th International Workshop on Edge Systems, Analytics and Networking (EdgeSys'21), April 26, 2021, Online, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3434770.3459738

1 INTRODUCTION

The *time value* of streaming data produced nowadays in the IoT is essential for *real-time* (or near real-time) control and automation applications. Data stream processing and analytics (DSPA) systems [16] are usually deployed in the Cloud where data are collected from different streams and are processed on the fly via a series of continuous operators such as aggregation, filter, join, etc. Such centralized solutions favor DSPA availability but may suffer from network congestion and delay issues in case of highly dynamic data streams. Additionally, data propagation to the Cloud may compromise privacy of sensitive data.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
EdgeSys'21, April 26, 2021, Online, United Kingdom
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8291-5/21/04…$15.00
https://doi.org/10.1145/3434770.3459738

Therefore it is beneficial to exploit the computational resources of an Edge-Fog-Cloud architecture [1] to partially aggregate and analyse data streams near their sources at the intermediate Edge and Fog layers and hence reduce network consumption and delay as well as enforce data privacy [12]. However, Edge/Fog layers have limited computational resource capacity and comes with energy consumption constraints. Thus, to distribute real-time applications at both Edge,Fog and Cloud layers, we need to consider optimal trade-offs between the management of network resources (e.g., bandwidth, delay) and the management of computational resources at often heterogeneous Edge/Fog layers [2, 20].

In this paper, we abstract the Edge-Fog-Cloud architecture as a hierarchical wide area resource network, where data streams produced by IoT devices at the Edge are transmitted to the closest Fog node while the locally processed data are transmitted to the central Cloud [1]. Unlike the Cloud, computational resources closer to the Edge are constrained [11] and their characteristics (memory, CPUs/GPUs) may change along with the mobility of the IoT devices generating the data streams. For these reasons to deploy a DSPA application across such architecture requires not only to decide an *optimal placement* of DSPA operators w.r.t. to both computational and network resources but also to *continuously monitor* the runtime conditions of their deployment in the Edge/Fog/Cloud.

Leveraging the computational resources of both the Fog and Cloud layers in real-time applications has been addressed in [11]. This work aims to minimize the task service delay considered as the sum of the computation delay, the queuing delay and network delay. In the contrary in our work we aim at minimizing both the Fog computational and Fog to Cloud network resource usage. On the other hand, related work on operator placement and scheduling mostly focus on optimizing network usage and ensuring system availability without optimizing computational resource usage [3, 9, 13–15, 17]. A centralized framework to place DSPA operators over a set of distributed nodes is proposed in [3]. This framework has as objective to optimize the overall response time of an application, the application availability and network resource usage while respecting constraints on the usage of the employed network and computational resources. This framework is extended in [9] to also include the enactment cost and the migration cost of continuous operators between different nodes by taking into account the heterogeneity of Fog resources. However, both previous solutions do not ensure that the usage cost of computational resource in the Fog is actually minimized. Moreover, SpanEdge [17] studies the placement of DSPA operators across the central (Cloud) and near-the-edge (Fog) data centers interconnected by a wide area network (WAN) in a decentralized manner. SpanEdge places operators near the Edge with the objective to minimize response time and network resource usage cost. Unfortunately, SpanEdge does not optimize the computational resource usage. On the other hand, Plannar [14] proposes a uniform approach for deploying a DSPA application between the

Cloud and the Edge. It relies on a minimum edge-cut algorithm to split separately each data stream processing path: the first part of the cut is deployed at the Edge by considering constraints involving data locality and computational resource usage, while the other is deployed on the Cloud. Plannar partitions aim to minimize both the network resource usage and the response time.

In response to the above-mentioned gaps, in this paper we made the following contributions:

- We introduce a holistic cost model that weights the usage of both computational and network resources of heterogeneous nodes in Edge-Fog-Cloud architectures;
- We propose two scheduling algorithms of DSPA operators over data streams at the network edge. The first, called RCS, *dynamically* places continuous operators from Cloud to Fog nodes while satisfying constraints on both network bandwidth and computational resource usage. The second, called SOO-CPLEX, *statically* places continuous operators on Fog and Cloud nodes by minimizing the combined cost of the computational and network resource usage while satisfying constraints on the usage of both types of resources;
- Both algorithms are thoroughly evaluated using IfogSim [8]. The simulation results show that SOO-CPLEX compared to RCS can achieve an overall resource usage cost reduction up to 37% for less costly Fog computational resources and up to 0.6% for costly Fog computational resources.

We organize the remainder of this paper as follows: Section 2 provides the resource allocation problem. Section 3 presents the scheduling algorithms. Section 4 describes the experimental evaluation. Section 5 presents the conclusion.

2 RESOURCE ALLOCATION PROBLEM

2.1 Preliminaries

DSPA application: is modeled as a directed acyclic graph of operators, denoted by G , where the vertices are the set of continuous operators and the edges are the set of data stream flowing between two operators [3]. G topology further includes the sources that produce the raw data streams S_j of rate $|S_j|$ consumed by DSPA operators and sinks that capture the stream of the computed results. To cope with the infinite nature of data streams, we consider that continuous operators are executed in time windows ω_i to process a finite set of data items arising within a time interval. Furthermore, we consider the following parameters:

Operator selectivity (sel_j) is the ratio between the input and output data rate of an operator O_j.

Cumulated operator selectivity ($csel_j$): is the product of operator selectivity from a source to a target operator O_j following the topological order of operators in the application graph G.

Edge data rate ($\lambda_{i,j}$) is the rate of data stream flow between two operators, from a source to an operator or from the application graph to a sink. This is calculated as the product of the cumulated operator selectivity of the upstream operator (or data source) and the rate of the raw data stream S_j: $\lambda_{i,j} = csel_i \cdot |S_j|$.

Edge-cut (ec_j) used in graph theory, is the partitioning of the application graph into two disjoints subgraphs. Any edge-cut contains a set of edges that have one endpoint in each subgraph of the partition. Hence let $|ec_j|$ denotes the value (or rate) of an edge-cut ec_j which is the sum of data rates of the edges crossing this edge-cut.

Minimum edge-cut is the edge-cut that has the smallest value among all the edge-cuts in the application graph G. To calculate the minimum edge-cut we may rely on the Edmond-Karp algorithm [7] that is proven to be efficient for dense graph.

Operator data load (ρ_j) is the aggregation of the input data streams per time window ω_j, we assume a specific window with ω_j=1sec:

$$\rho_j = \sum_{i=1}^{I} \omega_j \cdot \lambda_{i,j} \tag{1}$$

Where I is the maximum number of upstream operators O_i producing data stream at rate $\lambda_{i,j}$ towards the operator O_j.

Operator cost (c_j) is the computational cost in terms of cpu and/or memory usage for an operator O_j to process its data load ρ_j.

In the Edge-Fog-Cloud: we consider cm_{E_i} as the maximum cpu/memory of an Edge node E_i. The network link from an Edge node E_i to its nearest Fog node F_j is characterized by a network delay $nd_{E_iF_j}$ and a network bandwidth capacity nb_{EF_j} while cm_{F_j} denotes the maximum cpu/memory of Fog node F_j. The network link from a Fog node F_j to the Cloud node C is characterized by a network delay nd_{F_iC} and the maximum network bandwidth nb_{F_jC} while cm_C denotes the maximum cpu/memory of Cloud node C.

2.2 Computational resource usage cost

Our cost model extends [6] in order to take into account the usage cost of computational resources shared by multiple DSPA applications. More precisely, we weight the usage of a computational resource by an individual applications by the inverse of its computational capacity. The overall computational resource usage cost is:

$$cru = \sum_{i=1}^{M} cmu_{E_i} \cdot \frac{1}{cm_{E_i}} + \sum_{j=1}^{N} cmu_{F_j} \cdot \frac{1}{cm_{F_j}} + cmu_C \cdot \frac{1}{cm_C} \tag{2}$$

Where M is the total number of the Edge nodes E_i, N is the total number of the Fog nodes F_j, cmu_{E_i}, cmu_{F_j} and cmu_C are the cpu/memory usage respectively on the Edge node E_i, the Fog node F_j and the Cloud node C.

The more computational resources a node has the less is the cost for running a specific computation [11]. Thus, in the Cloud computational resources are practically infinite ($cm_C \to \infty$), so the weight is very small, practically zero ($\frac{1}{cm_C} \to 0$). In the Fog, computational resources are limited, so the weight is higher ($\frac{1}{cm_{F_j}} \to 1$) and the cost for using these resources is high, hence computational resources in the Fog should be used with parsimony as in addition it may have to be shared among multiple applications. We assume no processing is done at the Edge, hence the corresponding computational resources are not used, then formula (2) becomes:

$$cru = \sum_{j=1}^{N} cmu_{F_j} \cdot \frac{1}{cm_{F_j}} \tag{3}$$

cmu_{F_j} is the sum of cpu/memory usage required by each operator of the subgraph $Gmig_j \in G$, to replicate on the Fog node F_j:

$$cmu_{F_j} = \sum_{O_i \in Gmig_j} \rho_i \cdot c_i \tag{4}$$

2.3 Network resource usage cost

Unlike other works [5], we consider that modeling Edge-to-Fog network as a local-area network (LAN) and the Fog-to-Cloud network as a wide-area network (WAN), is too restrictive since a Fog node may be located at shorter but still considerable distance from the

Edge. Thus, in our cost model the Edge-Fog-Cloud is considered as a hierarchical wide area resource network. Hence, two conflicting factors are involved: network bandwidth increases up the hierarchy, however also network delay increases up the hierarchy. In the literature [3, 13], concerning peer node networks, network delay is used as the only weight factor for differentiating network links . In our case, network bandwidth should additionally be taken into account due to the hierarchical nature of the resource network. Then, the overall network resource usage cost is:

$$nru = \sum_{j=1}^{N}\sum_{i=1}^{M} nbu_{E_iF_j} \cdot \frac{1}{nb_{EF_j}} \cdot \frac{nd_{E_iF_j}}{nd_{min}} + \sum_{j=1}^{N} nbu_{F_jC} \cdot \frac{1}{nb_{FC}} \cdot \frac{nd_{F_jC}}{nd_{min}} \tag{5}$$

where M is the number of the Edge nodes E_i connected to their closest Fog node F_j, N is the number of the Fog nodes F_j. $nbu_{E_iF_j}$ and nbu_{F_jC} are respectively the network bandwidth usage on the Edge node E_i to Fog node F_j network link and on the Fog node F_j to the Cloud node C. nd_{min} is the minimum network delay in the resource network ($nd_{min} = \min_{i,j}\{nd_{E_iF_j}, nd_{F_jC}\}$), as the Edge is much closer to the Fog, normally one of the Edge-to-Fog network links has the minimum network delay.

Given that no processing is done at the Edge, the entire raw data stream S_j produced at the Edge reach the Fog anyway. Thus in the formula (5), the cost part concerning Edge-Fog network links is fixed (set as c constant). Furthermore if we assume that the network delay from Edge to Fog is the same for all the Edge-Fog network links, namely nd_{EF} and the network delay between any Fog node and the Cloud can be assumed equal namely nd_{FC}, then we can consider $\frac{nd_{FC}}{nd_{EF}}$ as a constant a. In this way Formula (5) becomes:

$$nru = c + a \cdot \sum_{j=1}^{N} nbu_{F_jC} \cdot \frac{1}{nb_{FC}} \tag{6}$$

The entire weight factor for a Fog-to-Cloud network link is the same for all the Fog-to-Cloud network links. Under the above assumptions, if we want to optimize *nru* alone without the constants *c* and *a* , *nru* can be simplified essentially as the total network bandwidth effectively used on all the Fog-to-Cloud network links:

$$B = \sum_{j=1}^{N} nbu_{F_jC} \tag{7}$$

2.4 Problem statement

Our goal is to design scheduling algorithm for DSPA operators that work in synergy with the evolution of IoT data stream rates so that we minimize the computational resource usage cost on the Fog nodes and the Fog-to-Cloud network resource usage cost. The problem is formally written as follows:

$$\text{minimize} \quad (cru, nru) \tag{8}$$

$$\text{subject to} \quad B \leq Bmax, \tag{9}$$

$$cmu_{F_j} \leq cm_{F_j} \; j = 1, \ldots, N, . \tag{10}$$

Where (9) is the constraint for the Fog-to-Cloud network bandwidth usage, we assume that the upper threshold *Bmax* is set for the specific DSPA application since Cloud providers may charge their clients based on network bandwidth usage additionally to the computational resource usage [4]. For distributed data intensive applications, the (monetary) cost of the former can be high. (10) is the constraint on computational resource of each Fog node F_j.

Furthermore, we consider the operator replicability constraint assuming that an operator can not be placed on a Fog node receiving local data stream while it supposes to process global data stream arriving to the Cloud, this is to ensure that the scheduling algorithm does not alter the semantic of a DSPA application.

Minimizing *cru* implies placement of the DSPA application in the Cloud which results in low Fog computational resource usage cost and high Fog-to-Cloud network resource usage cost; and minimizing *nru* implies placement of the DSPA application closer to the Edge, which results in high Fog computational resource usage cost and low Fog-to-Cloud network resource usage cost. In this respect there is a trade-off to address between the two metrics to guarantee a continuous optimal resource allocation.

3 SCHEDULING ALGORITHMS

Given a traffic monitoring system [19] where IoT data stream rates evolve randomly in different geographic areas, hence we first propose the scheduling algorithm RCS that regulates dynamically the computational and network resource usage to satisfy the constraints defined in 2.4. In second we propose the scheduling algorithm SOO-CPLEX that statically minimizes the overall computational and network resource usage cost and satisfies the constraints in 2.4, we plan to extend SOO-CPLEX to a dynamic approach in future work.

3.1 Resource constraint satisfaction

RCS is centralized (Algorithm 1), and we assume that the graph G of the DSPA application is initially deployed in the Cloud as all the data streams produced at the Edge arrive to the Cloud in anyway.

Algorithm 1: RCS

Input: G, application graph
Input: S, set of S_j arriving to the Cloud
Input: *B*, Fog-to-Cloud network bandwidth usage
Input: *Bmax*, Upper threshold for *B*
Input: *Bmin*, Lower threshold for *B*
Input: Grep ⊆ G, replicable subgraph in G
Input: *Fog*, set of Fog nodes F_j

```
if B > Bmax then
    ReplicateAndMigrateToFog()
else if B < Bmin then
    MigrateBackToCloud()
else if cmu_{F_j} > cm_{F_j} then
    AdjustEdgeCut()
```

We assume also that RCS monitors the Fog-to-Cloud network bandwidth usage and the Fog computational resource usage. In the following we describe how RCS reacts in synergy with the evolution of the data stream rates.

Replicate and migrate on the Fog: When $B > Bmax$, RCS triggers the function replicateAndMigrateToFog (Algorithm 2), the objective is to lower the Fog-to-Cloud network bandwidth usage *B* below the upper threshold *Bmax* while using as less as possible the Fog computational resources. In this respect RCS favors migrating the processing of high-rate raw data streams, which have a high impact on the Fog-to-Cloud network bandwidth usage. To this end, RCS sorts all the data streams S_j based on their rates, then RCS selects the highest-rate data stream S_j. For the selected S_j RCS identifies the maximum subgraph $Gsat_j \subseteq Grep$ that can be

migrated to the nearest Fog node F_j while satisfying the related computational resource constraint (10). RCS identifies the subgraph $Gmig_j \subseteq Gsat_j$ delimited by the minimum edge-cut of $Gsat_j$ and marks $Gmig_j$ as the part of the application graph G to replicate and migrate on the Fog. RCS updates B and checks if B is still above $Bmax$ in order to select the next highest-rate data stream. Otherwise RCS performs the reconfiguration of G.

Algorithm 2: RCS::replicateAndMigrateToFog

```
Function replicateAndMigrateToFog():
    Sort S in decreasing order
    Pick S_j on top of S if not yet migrated on the Fog
    Selected ← ∅∪S_j
    M ← ∅, set of subgraphs to deploy on the Fog
    while Selected ≠ ∅ do
        Pick S_j on top of Selected
        Select Fog node F_j receiving S_j
        Identify Gsat_j ⊆ Grep while cmu_Fj ≤ cm_Fj
        Find minimum edge-cut ec_j in Gsat_j
        Find Gmig_j ⊆ Gsat_j delimited by ec_j
        M[j] ← Gmig_j
        B ← B − |S_j| + |ec_j|
        if B > Bmax then
            Pick S_j on top of S if not yet migrated on the Fog
            Selected ← Selected ∪S_j
    Rewrite G to include all Gmig_j ∈ M
    Deploy the new G
    Redirect all selected S_j to be processed on the Fog
```

Migrate back to the Cloud: When RCS has replicated a part of DSPA application on the Fog, we need a way to move the processing of the data streams back to the Cloud when using the Fog computational resources is not necessary any more. Thus we set a lower threshold $Bmin$, when B gets lower than $Bmin$, the rates of some data streams S_j arriving to the Fog decrease so that it is no longer necessary to use the Fog resources as before B became less than $Bmin$. In this case RCS triggers the function migrateBackToCloud

Algorithm 3: RCS::migrateBackToCloud

```
Function migrateBackToCloud():
    Sort S in increasing order
    Pick S_j on top of S, if migrated on Fog
    Selected ← ∅∪S_j
    R set of Gmig_j to remove on the Fog
    while Selected ≠ ∅ do
        Pick S_j on top of Selected
        ec_j ← MR[j]
        Gmig_j ← M[j]
        if (B + |S_j| − |ec_j|) < Bmax then
            B ← B + |S_j| − |ec_j|
            R[j] ← Gmig_j
            Pick S_j on top of S, if migrated on Fog
            Selected ← Selected ∪S_j
    Rewrite G to remove all Gmig_j ∈ R
    Deploy the new G
    Redirect all selected S_j to be processed in Cloud
```

(Algorithm 3), the objective is to raise B as much as possible above the lower threshold $Bmin$ by satisfying the constraint $B \leq Bmax$. In this respect, RCS favors migrating back to Cloud, the data streams, previously migrated on the Fog, that has the lowest negative impact on the constraint $B \leq Bmax$. To this end, RCS sorts all the data streams S_j migrated on the Fog, then selects the lowest-rate data stream S_j migrated on the Fog. Based on the selected data stream S_j, RCS identifies $Gmig_j$ to be removed from the Fog, checks if removing $Gmig_j$ will still keep B lower than the upper threshold $Bmax$. If this is true, RCS actually updates B and mark $Gmig_j$ to be removed and selects the next lowest-rate data stream S_j. Otherwise RCS performs the reconfiguration of G to remove all marked subgraphs $Gmig_j$ and redirect the corresponding data streams to be processed directly in the Cloud.

Adjust edge-cut: When $cmu_{F_j} > cm_{F_j}$ for one of the Fog node F_j on which a subgraph $Gmig_j$ is deployed, RCS triggers the function AdjustEdgeCut (Algorithm 8). The objective is to readjust the current subgraph $Gmig_j$ so that the resulting cmu_{F_j} satisfies the constraint $cmu_{F_j} \leq cm_{F_j}$.

Algorithm 3: RCS::adjustEdgeCut

```
Function adjustEdgeCut():
    Get current S_j served by F_j
    Gmig_current ← M[j]
    Get Gsat_j ⊆ Gmig_current where cmu_Fj ≤ cm_Fj
    Find minimum ec_j in Gsat_j
    Find Gmig_j ⊆ Gsat_j delimited by ec_j
    Rewrite G to replace Gmig_current by Gmig_j
    Deploy the new G
```

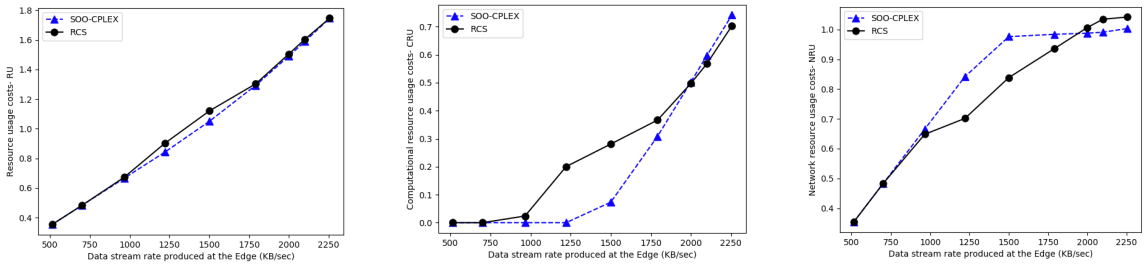
In this respect we identify the data stream S_j that is served by the Fog node F_j, then we identify the subgraph $Gmig_{current}$ which is currently deployed on this Fog node F_j. Based on $Gmig_{current}$ and the rate of the data stream S_j, we identify the subgraph $Gsat_j \subseteq Gmig_{current}$ that satisfy the computational resource constraint of the Fog node F_j, then we select the minimum edge-cut ec_j in $Gsat_j$ based on which we identify the replicable subgraph $Gmig_j \subseteq Gsat$ to replace $Gmig_{current}$. Upon identifying the new $Gmig_j$, we perform the reconfiguration of the application graph G to replace the current $Gmig_{current}$ by $Gmig_j$ on the Fog node F_j.

Rewriting the graph G: is about to transform G into an equivalent one by replicating a part or removing an already replicated part of G by using query rewriting as a DSPA operator is a query [10].

3.2 Single-objective optimization

RCS satisfies only the constraints defined in our problem (Section 2.4). Even though RCS uses as less as possible the Fog computational resources. However RCS is oblivious to the minimization of the costs of using the computational and network resources.

In this respect we transform our problem (Section 2.4) into a single-objective optimization (SOO) problem by applying weights to the different optimization metrics. The SOO problem is potentially combinatorial [3], thus we use the integer linear programming (ILP) model. To this end, we introduce a decision variable X_{jk} to set an edge-cut $ec_k \in Grep$ as the replication and migration point of a data stream S_j to be processed on the Fog node F_j. X_{jk} is equal to 1 if ec_k is set to a data stream S_j, otherwise X_{jk} is equal to 0. We



(a) Overall resource usage cost (b) Computational resource usage cost (c) Network resource usage cost

Figure 1: Results with costly Fog computational resources

consider the subgraph $Gmig_j \in Grep$ delimited by the edge-cut ec_k as the candidate subgraph to replicate and migrate on the Fog node F_j to process the data stream S_j. Then the computational resource usage cost defined in (3) becomes:

$$cru = \sum_{j=1}^{N} \sum_{k=1}^{K} cmu_{F_{jk}} \cdot \frac{1}{cm_{F_j}} \cdot X_{jk} \tag{11}$$

The network resource usage cost defined in (6) becomes:

$$nru = c + a \cdot \sum_{j=1}^{N} \sum_{k=1}^{K} nbu_{F_jC_k} \cdot \frac{1}{nb_{FC}} \cdot X_{jk} \tag{12}$$

Where N is the maximum number of Fog nodes F_j and K the maximum number of edge-cuts identified in the replicable subgraph $Grep \subset G$. $cmu_{F_{jk}}$ is the computational resource usage for a candidate subgraph $Gmig_j$ delimited by the edge-cut ec_k on the Fog node F_j. $nbu_{F_jC_k}$ is the Fog-to-Cloud network bandwidth usage from the Fog node F_j to the Cloud node C when the edge-cut ec_k is considered as the migration and replication point in Grep.

Since the metrics cru and nru are defined in different scales, we normalize these two metrics by using the min-max scaling technique, then cru defined in the Formula (11) becomes:

$$CRU = \frac{cru - cru_{min}}{cru_{max} - cru_{min}} \tag{13}$$

cru_{min} is the minimum value set to 0 and cru_{max} is the maximum value set to the number of Fog nodes.

Furthermore, we normalize nru defined in Formula (12) as following:

$$NRU = \frac{nru - cru_{min}}{nru_{max} - nru_{min}} \tag{14}$$

Where nru_{min} is the minimum value set to 0 and nru_{max} is the maximum value set to 1 as all the Fog nodes share a unique network bandwidth to access to the Cloud.

Then the objective is to minimize the overall resource usage cost:

$$\text{minimize} \quad RU = wc \cdot CRU + wn \cdot NRU \tag{15}$$

$$\text{subject to} \quad \sum_{k=1}^{K} cmu_{F_{jk}} \cdot X_{jk} \leq cm_{F_j} \, j = 1, \ldots, N, \tag{16}$$

$$\sum_{j=1}^{N} \sum_{k=1}^{K} nbu_{F_{jkC}} \cdot X_{jk} \leq Bmax \tag{17}$$

$$\sum_{k=1}^{K} X_{jk} = 1, j = 1, \ldots, N. \tag{18}$$

Where w_c and w_n are respectively the weight for the computational and network resource usage cost, in the rest of the paper $w_c = w_n = 1$. Formula (16) is the constraint of the computational resource usage of each individual Fog node F_j, and Formula (17) is the constraint of the Fog-to-Cloud network bandwidth usage and Formula (18) is the constraint that requires to set only one edge-cut ec_k as the replication and migration point in the subgraph Grep for a data stream S_j. For simplicity, we use SOO-CPLEX to identify our SOO problem as we rely on the CPLEX tool[18] to solve this problem.

4 EXPERIMENTAL EVALUATION

We use the iFogSim simulator in which we implement the algorithms RCS and SOO-CPLEX using Java, to solve the SOO-CPLEX we use the CPLEX tool. We set in iFogSim the Edge-Fog-Cloud containing 4 Fog nodes and 1 Cloud node and a set of data source cluster at the Edge that produce per cluster the data streams S_1, S_2, S_3 and S_4 respectively towards the Fog nodes F_1, F_2, F_3 and F_4. All the Fog nodes are connected to the Cloud node. We then set in iFogSim the DSPA application in Figure 2, so that the cumulated operator selectivity is decreasing and the cumulated operator cost is increasing as we go from the source to the sink. For each operator O_i, $sel_i < 1$; $c_i > 1$.

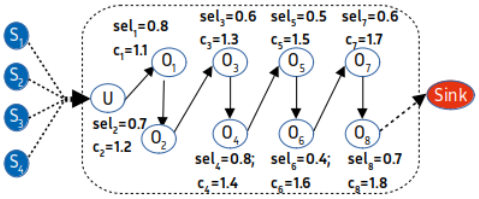
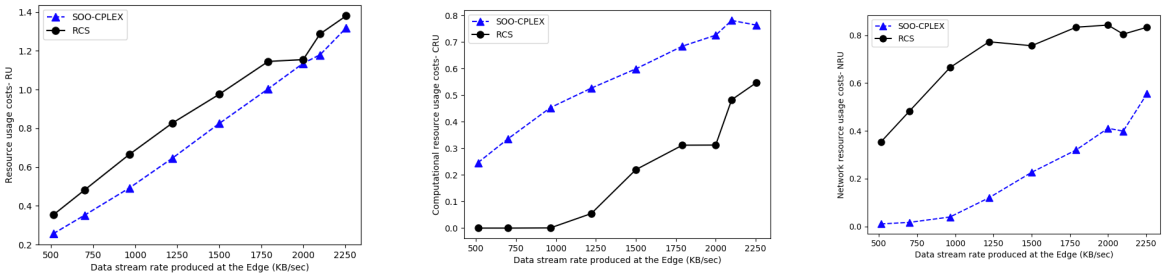


Figure 2: DSPA application used in the evaluation

To simulate the random behaviour of the rates of the data streams S_1,S_2,S_3 and S_4, we create arbitrarily a set of 9 total data stream rates in the interval [512KB/sec, 2256KB/sec]. For each total data stream rate we set an interval [0, total data stream rate] in which we randomly (uniformly) distribute the rate of each of the data stream, so that the sum-rate is equal to the total data stream rate. We wish to have around 15 results of the resource usage costs for each total data stream rate and plot the average of these results per total data stream rate. In this respect, we create a suite of 135 (9*15) total data stream rates that evolve randomly, that we feed to RCS and SOO-CPLEX. For RCS, we maintain the reconfiguration state of the DSPA application between the successive experiments but for SOO-CPLEX, each experiment is independent.

To evaluate RCS in the 3 cases, we set $Bmin = 800KB/sec$ and $Bmax = 1450KB/sec$ so that we have at least 2 total data stream rates lower than $Bmin$ and 2 others between $Bmin$ and $Bmax$. Furthermore the evaluation is carried in two cases: with costly Fog computational resources in terms of memory, we set $cm_{F_j} = 1280KB$ for each Fog node; with less costly Fog computational resources in terms of memory, we set $cm_{F_j} = 2048KB$ for each Fog node.



(a) Overall resource usage cost

(b) Computational resource usage cost

(c) Network resource usage cost

Figure 3: Results with less costly Fog computational resources

4.1 Evaluation results

We present the evaluation results of the metrics *RU*, *CRU* and *NRU* of RCS and SOO-CPLEX. In this respect we consider as modest data stream rates, the data stream with rates lower than *Bmax* (512KB/sec to 1221KB/sec) and as higher data stream rates, the data stream with rates higher than *Bmax* (1500KB/sec to 2256KB/sec).

Costly Fog computational resources: At modest data stream rates, Figure 1a shows that RCS performs like SOO-CPLEX for data stream rates lower than *Bmin*. Since $B \leq Bmin$ (i.e. $NRU \leq 0.55$ in Figure 1c), RCS has the preference to not use the Fog computational resources or to release the latter if a part of the DSPA application was migrated on. For the remaining modest data stream rates, SOO-CPLEX has a lower *RU*, hence outperforms RCS with a gap up to 0.6%. At higher data stream rates, *RU* for SOO-CPLEX outperforms RCS, since RCS has higher *RU* with an increasing ratio up to 0.6%. However as long as the data stream rate increase, the difference between SOO and RCS is getting smaller which is owing to the fact that on one hand RCS is using more and more Fog computational resources to satisfy the constraint $B \leq Bmax$ without minimizing *RU*; and on the other hand since the Fog computational resources are costly, then SOO-CPLEX for minimizing *RU* and satisfying the constraint $B \leq Bmax$, uses more Fog computational resources. Figures 1b and 1c show that SOO-CPLEX balances *CRU* and *NRU*, since at modest data stream rates it minimizes *CRU* so that its value is null and constant while *NRU* is monotonically increasing up to 0.84 as long as the modest data stream rates are lower than *Bmax*. However for higher data stream rates, the behaviour of SOO-CPLEX is reversed, *CRU* is increasing monotonically but in small proportions up to 0.74 while *NRU* has high values up to 0.9 but keep constant in the form of a tray, until the Fog computational resources are no longer sufficient to serve the maximum data stream rate 2256KB/sec, then $NRU > 1.0$.

Less costly Fog computational resources: Figure 3a shows that whatever the evolution of the data stream rates, SOO-CPLEX outperforms RCS with a gap up to 37% at modest data stream rates and up to 18% at higher data stream rates. Figures 3b and 3c show that SOO-CPLEX finds a certain balance between the overall computational resource usage cost *CRU* and the overall network resource usage cost *NRU* to minimize the overall resource usage cost *RU*. Regarding RCS, at modest data stream rates lower than *Bmin*, no subgraph $Gmig_j$ is replicated on the Fog, hence *CRU* is null, such cost can be cost-efficient, however the counterpart in terms of *NRU* is very high comparing to SOO-CPLEX. Then at high data stream rates, as the computational resources are less coslty and the cumulated operator selectivities are monotonically decreasing in the DSPA application (Figure 2), RCS finds the minimum edge-cut (with lowest data rates) closer to the sink of the DSPA application, that decreases drastically *B* so that RCS satisfies the constraint $B \leq Bmax$ by replicating on the Fog a small number of subgraph $Gmig_j$. However *B* (consequently *NRU*) is very high when comparing to SOO-CPLEX.

5 CONCLUSIONS

In this paper we addressed the problem of scheduling continuous operators between the Cloud and the Fog in order to cope with highly dynamic IoT data stream produced at the Edge. To this end we proposed two scheduling algorithms RCS and SOO-CPLEX that take into account the limited Fog computational resources and congestion and delay issues on Fog-to-Cloud network resources. Evaluation results demonstrated that SOO-CPLEX achieves always an optimal overall resource usage cost due to an optimal trade-off between the Fog computational resource usage cost and the Fog-to-Cloud network resource usage cost. However, as SOO-CPLEX is based on ILP optimization, it may raise scalability concerns for large scale instance of the problem. We are currently working on a heuristic optimization framework that approximates optimal SOO-CPLEX solutions and also accounts for dynamic resource optimization by continuously monitoring resources' usage.

REFERENCES

[1] Luiz Bittencourt, Roger Immich, Rizos Sakellariou, Nelson Fonseca, Edmundo Madeira, et al. 2018. The internet of things, fog and cloud continuum: Integration and challenges. *Internet of Things* (2018).
[2] Antonio Brogi et al. 2017. QoS-aware deployment of IoT applications through the fog. *IEEE Internet of Things Journal* (2017).
[3] Valeria Cardellini, Vincenzo Grassi, Francesco Lo Presti, and Matteo Nardelli. 2016. Optimal operator placement for distributed stream processing applications. In *10th ACM DEBS*.
[4] Marcos Dias De Assunção, Alexandre Di Costanzo, and Rajkumar Buyya. 2009. Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. In *Proceedings of the 18th ACM HPDC*.
[5] Ruilong Deng, Rongxing Lu, Chengzhe Lai, Tom H Luan, and Hao Liang. 2016. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. *IEEE internet of things journal* (2016).
[6] Bruno Donassolo, Ilhem Fajjari, Arnaud Legrand, and Panayotis Mertikopoulos. 2019. Fog based framework for IoT service provisioning. In *2019 16th CCNC*.
[7] Jack Edmonds and Richard M Karp. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)* (1972).
[8] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K Ghosh, and Rajkumar Buyya. 2017. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software: Practice and Experience* (2017).
[9] Thomas Hiessl, Vasileios Karagiannis, Christoph Hochreiner, Stefan Schulte, and Matteo Nardelli. 2019. Optimal placement of stream processing operators in the fog. In *2019 IEEE 3rd ICFEC*.

[10] Albert Jonathan, Abhishek Chandra, and Jon Weissman. 2020. WASP: Wide-area Adaptive Stream Processing. In *Proceedings of the 21st ACM/IFIP*.
[11] Lei Li, Mian Guo, et al. 2019. Online workload allocation via fog-fog-cloud cooperation to reduce IoT task service delay. *Sensors* (2019).
[12] Jonathan McChesney et al. 2019. Defog: fog computing benchmarks. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*.
[13] Peter Pietzuch, Jonathan Ledlie, Jeffrey Shneidman, Mema Roussopoulos, Matt Welsh, and Margo Seltzer. 2006. Network-aware operator placement for stream-processing systems. In *22nd IEEE ICDE'06*.
[14] Laurent Prosperi, Alexandru Costan, Pedro Silva, and Gabriel Antoniu. 2018. Planner: cost-efficient execution plans placement for uniform stream analytics on edge and cloud. In *2018 IEEE/ACM WORKS*.
[15] Stamatia Rizou et al. 2010. Solving the multi-operator placement problem in large-scale operator networks. In *2010 Proceedings of 19th IEEE ICCCN*.
[16] Henriette Röger and Ruben Mayer. 2019. A comprehensive survey on parallelization and elasticity in stream processing. *ACM CSUR* (2019).
[17] Hooman Peiro Sajjad, Ken Danniswara, Ahmad Al-Shishtawy, and Vladimir Vlassov. 2016. Spanedge: Towards unifying stream processing over central and near-the-edge data centers. In *2016 IEEE/ACM SEC*.
[18] IBM ILOG CPLEX Optimization Studio-CPLEX. 2019. Users manual-version 12 release 8. *IBM ILOG CPLEX Division: Incline Village, NV, USA* (2019).
[19] Abhishek Tiwari, Brian Ramprasad, Seyed Hossein Mortazavi, et al. 2019. Reconfigurable Streaming for the Mobile Edge. In *20th HotMobile*.
[20] Luis M Vaquero et al. 2014. Finding your way in the fog: Towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review* (2014).